Computer Science Journal of Moldova, vol.**22**, no.1(64), 2014

Dynamic Object Identification with SOM-based neural networks

Aleksey Averkin, Veaceslav Albu, Sergey Ulyanov, Ilya Povidalo

Abstract

In this article a number of neural networks based on self-organizing maps, that can be successfully used for dynamic object identification, is described. Unique SOM-based modular neural networks with vector quantized associative memory and recurrent self-organizing maps as modules are presented. The structured algorithms of learning and operation of such SOM-based neural networks are described in details, also some experimental results and comparison with some other neural networks are given.

Keywords: Neural networks; forecasting; timeseries prediction; dynamic object identification.

1 Introduction

Identification theory solves problems of constructing mathematical models of dynamic systems according to the observations of their behaviour. The object identification step is one of the most important steps while constructing mathematical models of objects or processes. The quality of the model relies on this step and, therefore, the quality of control, that is based on this model, or results of a research with this model also rely on this step.

Dynamic object identification is one of the basic problems which could be solved using neural networks [1] along with different other methods. Object identification is complicated if noises are present in the source data, some of the object parameters change according to unknown laws or the exact number of the object parameters is unknown.

In such cases neural network can be applied for dynamic object identification. There are a lot of different types of neural networks that can be used for dynamic object identification.

Among different neural network architectures applicable for dynamic object identification a class of neural networks based on self-organizing maps (SOM) can be noted. Neural networks of such type will be given special attention in this article due to their wider spread and successful application in solving different kinds of problems [2,3] including problems of forecasting and identification. A number of biomorphic neural networks, architecture of which is the result of studying the structure of the cerebral cortex of mammals, will also be considered.

2 SOM-based neural networks that can be used for dynamic object identification

2.1 Problem definition

Identification of a dynamic object that receives a vector of input parameters $u(t)$ at time t and gives an output vector $y(t)$ can be described as finding the type of a model of this object, that has an output $\hat{y}(t)$ and finding parameters of this model such that minimize error $e = \| y(t) - \hat{y}(t) \|^2$ of this model (see figure 1).

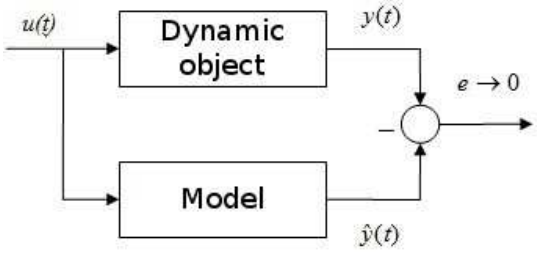

Figure 1. Dynamic object identification scheme

Suppose that there is a sequence of vectors of input parameters $u(t)$, $t\epsilon[0,T]$ and a sequence of output vectors $y(t)$, $t\epsilon[0,T]$, where T

– number of input-output pairs. We will consider the solution of the object identification problem as definition of the type of the function f that will define the model of identified object:

$$\hat{y}(t) = f[y(t-1), ..., y(t-n_y), u(t), u(t-1), ..., u(t-n_u)], \quad (1)$$

where $\hat{y}(t)$ is vector of output parameters of the model. At a single moment of time the input of the model takes current known measured values of the input parameters along with $n_u < T$ previous values and $n_y < T$ previous output parameters of the identified object.

Also we will consider solution of this problem in the following form:

$$\hat{y}(t) = f[\hat{y}(t-1), ..., \hat{y}(t-n_y), u(t), u(t-1), ..., u(t-n_u)], \quad (2)$$

where $\hat{y}(0) = y(0)$, $\hat{y}(t)$ is vector of output parameters of the model, $n_u < T$. This identification scheme has recurrent connections and at a single moment of time the input of the model takes current known measured values of the input parameters of the object along with $n_u < T$ previous values of these parameters and $n_y < T$ previous output parameters of the model.

2.2 Vector quantized temporal associative memory (VQTAM)

VQTAM is a modification of self-organizing maps which can be used for identification of dynamic objects [4,5]. The input vector $u(t)$ of this network is split into two parts: $x^{in}(t)$, $x^{out}(t)$. The first part of the input vector $x^{in}(t)$ contains information about the inputs of the dynamic object and its outputs at previous time steps. The second part of the input vector $x^{out}(t)$ contains information about the expected output of the dynamic object corresponding to the input $x^{in}(t)$. The weights vector is also split into two parts in a similar way [4]. Thus $x(t) = \begin{pmatrix} x^{in}(t) \\ x^{out}(t) \end{pmatrix}$ and $w_i(t) = \begin{pmatrix} w_i^{in}(t) \\ w_i^{out}(t) \end{pmatrix}$, where $w_i(t)$ is weights vector of the i-th neuron, $w_i^{in}(t)$ is the part of the weights vector that contains information on the inputs, and $w_i^{out}(t)$ is the part of the weights vector

that contains information on the outputs. The first part of the input vector contains information on the process inputs and its previous outputs:

$$x^{in}(t) = (y(t-1), ..., y(t-n_y), u(t), u(t-1), ..., u(t-n_u)), \quad (3)$$

where $n_u < T$, $n_y < T$. The second part of the input vector $x^{out} = y(t)$ contains information on the expected output of the process corresponding to the inputs $x^in(t)$.

Each vector in the learning sample consists of a pair of vectors $(y(t), u(t))$ and the sample should contain not less than $max(n_u, n_y)$ vectors. Vectors $y(t)$ are the measured output parameters of the process at time step t, and $u(t)$ are the input parameters of the process at the same time step.

After presenting a subsequent input vector $x(t)$, combined of several vectors from the learning sample, to the network the winner neuron is determined only by the $x^{in}(t)$ part of the vector:

$$i^*(t) = \arg\min_i \left\{ \| \, x^{in}(t) - w_i^{in}(t) \, \| \right\}, \quad (4)$$

where $i^*(t)$ is a number of the winner neuron at time step t.

For weight modification a modified SOM weight modification rule is used:

$$\begin{aligned} \Delta w_i^{in}(t) &= \alpha(t) h(i^*, i, t)[x^{in}(t) - x_i^{in}(t)], \\ \Delta w_i^{out}(t) &= \alpha(t) h(i^*, i, t)[x^{out}(t) - w_i^{out}(t)], \end{aligned} \quad (5)$$

where $0 < \alpha(t) < 1$ is a learning rate of the network, h is neighbourhood function of the i-th and i^*-th neurons. For example a Gaussian function can be chosen as a neighbourhood function $h(i^*, i, t)$:

$$h(i^*, i, t) = \exp\left(-\frac{\| \, r_i(t) - r_{i^*}(t) \, \|^2}{2\sigma^2(t)} \right), \quad (6)$$

where $r_i(t)$ and $r_{i^*}(t)$ are positions on the map of the i-th and i^*-th neurons, $\sigma(t) > 0$ determines the radius of the neighbourhood function at time step t. When the winner neuron i^* is defined, the output of the network is set to $w_{i^*}^{out}(t)$.

On the test sample VQTAM's input takes only $x^{in}(t)$ part of the input, which is used to define the winner neuron, and the output of the network is set to $w_{i^*}^{out}(t)$. Vector $w_{i^*}^{out}(t)$ can be interpreted as the predicted output $\hat{y}(t)$ of the dynamic object at time step t.

Learning algorithm of VQTAM network is similar to regular SOM algorithm:

1. Weights are initialized with random values or values from the training sample.

2. A vector from the sample is presented to the network and the winner neuron is identified according to equation 4.

3. Weights are modified according to the rule 5.

4. Steps 2 and 3 are repeated for each vector from the sample.

5. Steps 2 – 4 are repeated for the sample several times until a specified number of epochs has passed or a desired accuracy of identification has been reached.

On a test sample after presenting an input vector containing only the first $x^{in}(t)$ part of the input vector a winner neuron is identified and the output of the network (modeled output of the object) is set equal to the second part of the weight vector of the winner neuron $w_{i^*}^{out}(t)$. The output can be also identified as average between several $w_i^{out}(t)$ of best-matching neurons.

2.3 Recurrent self-organizing map (RSOM)

In RSOM unlike conventional SOMs with recurrent connections, a decaying in time vector of outputs is introduced for each neuron. This vector is used to determine the winner neuron and is used in maps weights modification [6, 7].

The network inputs vector $x(t)$ is represented as follows:

$$x(t) = (y(t-1), ..., y(t-n_y), u(t), u(t-1), ..., u(t-n_u))), \quad (7)$$

where $n_u < T$, $n_y < T$.

Output of each neuron is determined according to the following equation:

$$V_i(t) = \| \nu_i(t) \|, \tag{8}$$

where $\nu_i(t) = (1-\alpha)\nu_i(t-1) + \alpha(x(t) - w_i(t))$, α is the output decay factor ($0 < \alpha \leqslant 1$), $V_i(t)$ is output of the i-th neuron at time step t, $w_i(t)$ is weights vector of the i-th neuron. Further in the article it will be shown, that a neuron with defined like this output is close to definition of a chaotic neuron, and also some benefits of this approach will be described.

After presenting a subsequent input vector to the network a winner neuron is determined as the neuron with minimal output [7]:

$$i^*(t) = \arg\min_i \{V_i(t)\}. \tag{9}$$

To modify the weights a modified conventional SOM rule is used:

$$\Delta w_i(t) = \alpha(n)h(i^*, i, t)\nu_i(t), \tag{10}$$

where $0 < \alpha(t) < 1$ is learning rate, h is neighbourhood function of i-th and i^*-th neurons.

When the learning process is complete, the network is presented again with the learning sample and clusterizes it. Each cluster can be approximated with an individual model, for example a linear function $f_i(t)$ for the i-th cluster. Thus, after presenting the learning sample a linear function is defined for each vector of this sample. These functions can be used to determine the output value at the next time step.

This process can be speeded up by using algorithms for constructing local linear models while training the neural network. Each neuron of the RSOM network is associated with a matrix $A_i(t)$ that contains coefficients of the corresponding linear model:

$$A_i(t) = \left[b_{i,1}(t), ..., b_{i,n_y}(t), a_{i,1}(t), ..., a_{i,n_u}(t)\right]^T, \tag{11}$$

The output value of the network is defined according to the following equation:

$$\hat{y}(t) = \sum_{k=1}^{n_y} b_{i^*,k}(t)u(t-k) + \sum_{l=1}^{n_u} a_{i^*,i}(t)y(t-l) = A_{i^*}^T(t)x(t), \qquad (12)$$

where $A_{i^*}(t)$ is the matrix of coefficients associated with the winner neuron $i^*(t)$. Matrix $A_{i^*}(t)$ is used for linear approximation of model's output.

While constructing the local linear models simultaneously with training of the neural network an additional rule to modify the coefficients of the linear model is needed:

$$A_i(t+1) = A_i(t) + \beta h(i^*, i, t)\Delta A_i(t), \qquad (13)$$

where $0 < \beta < 1$ is learning rate of the model, $\Delta A_i(t)$ is Widrow-Hoff's rule for error correction:

$$\Delta A_i(t) = \left[y(t) - A_i^T(t)x(t)\right] \frac{x(t)}{\parallel x(t) \parallel^2}, \qquad (14)$$

where $y(t)$ is desired output of the model for the $x(t)$ input.

Thus, at each step of the network training a modification of the model coefficients is performed along with modification of the weights of neurons. On the test sample after an input vector is presented to the network a winner neuron $i^*(t)$ is chosen. Then a corresponding coefficients matrix $A_{i^*}(t)$ of the linear model is calculated. Using the determined matrix the output of the model is defined by the equation: $\hat{y}(t) = A_{i^*}^T(t)x(t)$.

So, learning algorithm of RSOM network is similar to SOM training algorithm but with some differences:

1. Weights are initialized with random values or values from the training sample.

2. Parameters of the local models assigned to neurons are initialized with random values.

3. A vector from the sample is presented to the network, outputs of all neurons are calculated according to (8) and the winner neuron is identified by (9).

4. Weights are modified according to the rule (10).

5. Local models parameters are modified according to (13).

6. Steps 3 – 5 are repeated for each vector from the sample.

7. Steps 3 – 6 are repeated for the sample several times until a specified number of epochs has passed or a desired accuracy of identification has been reached.

On a test sample after presenting an input vector, outputs of all neurons are calculated according to (8), the winner neuron is identified and the corresponding local model is chosen and the resulting modelled output is determined by equation (12).

2.4 Modular self-organizing maps

Modular self-organizing maps are presented in Tetsuo Furukava's works [8, 9]. Modular SOM has a structure of an array which consists of functional modules that are actually trainable neural networks (see figure 2), such as multilayer perceptrons (MLP), but not a vector, as in conventional self-organizing maps. In case of MLP-modules modular self-organizing map finds features or correlations in input and output values simultaneously building a map of their similarity. Thus, a modular self-organizing map with MLP modules is a self-organizing map in a function space but not in a vector space [9].

These neural network structures can be considered biomorphic, as their emergence is due to research of the brain structure of mammals [10], and confirmed by a number of further studies [11]. The basis of the idea of the cerebral cortex structure is a model of cellular structure, where each cell is a collection of neurons, a neural column. Columns of neurons are combined in a more complex structures. In this regard it was suggested to model the individual neural columns with neural

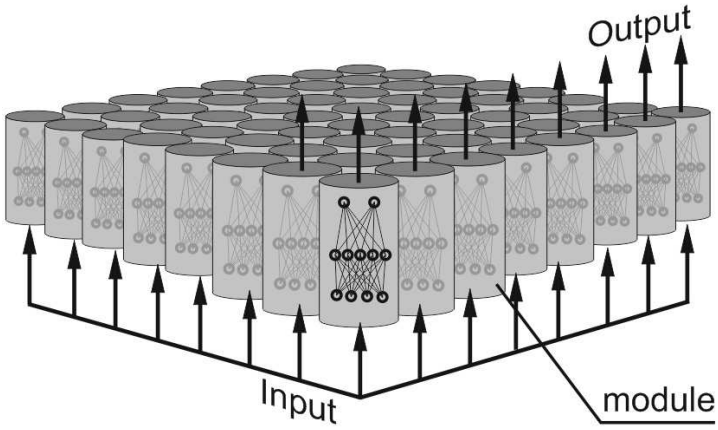

Figure 2. Modular network structure

networks [11]. This idea has formed the basis of the modular neural networks.

In fact, the modular self-organizing map is a common SOM, where neurons are replaced by more complex and autonomous entities such as other neural networks. Such replacement requires a slight modification of the learning algorithm. In the proposed by [9] algorithm at the initial stage the network receives the i-th sample of the input data corresponding to I functions, which will be mapped by the network, and the error is calculated for each network module:

$$E_i^k = \frac{1}{J}\sum_{j=1}^{J} \parallel \hat{y}_{i,j}^k - y_{i,j} \parallel^2, \tag{15}$$

where k is module number, for which the error is calculated, J is number of vectors in the sample, $\hat{y}_j^k$ is output of the k-th module, y_j is expected output of the network on the suggested set of input data. Winner module is calculated as the module that minimizes the error E^k:

$$k_i^* = \arg\min_k E_i^k. \tag{16}$$

As soon as the winner module is defined, the adaptation process

takes place and the weights of the module are being modified according to one of learning algorithms suitable for the networks of this type, after that the weights of the main SOM are being modified. In this process parameters (weights) of each module are considered as the weights of the SOM and are modified according to standard learning algorithms suitable for conventional SOMs.

In this study SOMxVQTAM and SOMxRSOM networks were developed, which are SOMs with modules of VQTAM type and SOMs with modules of RSOM type respectively. Further some application results of such networks will be reviewed.

3 Using SOM-based neural networks for dynamic object identification

For some experiments and comparisons of the algorithms the neural networks of types VQTAM, RSOM, SOMxVQTAM and SOMxRSOM were tested on samples that were used in 2006 – 2007 to identify the winners at neural networks forecasting competition [14]. Results of these competitions were used in this study as there is a detailed description of the place definition method used for all competitors. Also a fair amount of different algorithms took part in this competion and there was a description for the most of those algorithms as well as the learning and testing samples, which allowed a comparative analysis of the neural networks described in this paper with other advanced algorithms. To determine the place in the table, the organizers of the competition [14] suggested to forecast 18 steps for each of the 111 samples. For each of the resulting predictions a symmetric mean absolute percentage error ($SMAPE$) was calculated:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{\parallel y(t) - \hat{y}(t) \parallel}{(y(t) + \hat{y}(t))/2} * 100, \tag{17}$$

where $y(t)$ is the real state of the object at time step t, $\hat{y}(t)$ is the output of the model at time step t, n is the number of vectors in the

testing sample (18 for this competition). Then the place in the table was determined from the average error for all of 111 samples.

Each of the 111 samples had different lengths (from 51 to 126 points in the training samples), those samples represented a monthly measure of several macroeconomic indicators.

Each of the algorithms described in this article were launched with the same parameters for all of 111 samples, that is the parameters for each network were set only once before presenting the set of all 111 samples, but not set individually for each of the 111 samples, which lead to some not very successful forecasts that led to growth of the average error. Despite this fact algorithms could accurately predict the future values, as it is clearly seen from the results table (see Table 1). The $SMAPE$ error of the forecast for most of the 111 samples was lower than 5% (for more than 70% of all samples) but the average error grew due to unsuccessful forecasts with error values up to 60%. The above error can be greatly reduced if the automatic tuning of network parameters for each of the samples would be applied.

Table 1: Results table for different forecasting algorithms

Num.	Algorithm name	SMAPE
1	Stat. Contender - Wildi	14,84%
2	Stat. Benchmark - Theta Method (Nikolopoulos)	14,89%
3	Illies, Jager, Kosuchinas, Rincon, Sakenas, Vaskevcius	15,18%
4	Stat. Benchmark - ForecastPro (Stellwagen)	15,44%
5	CI Benchmark - Theta AI (Nikolopoulos)	15,66%
6	Stat. Benchmark - Autobox (Reilly)	15,95%
7	Adeodato, Vasconcelos, Arnaud, Chunha, Monteiro	16,17%
8	Flores, Anaya, Ramirez, Morales	16,31%
9	Chen, Yao	16,55%
10	D'yakonov	16,57%
11	Kamel, Atiya, Gayar, El-Shishiny	16,92%

Continuation of Table 1

Num.	Algorithm name	SMAPE
12	Abou-Nasr	17,54%
13	Theodosiou, Swamy	17,55%
–	**VQTAM**	17,61%
–	**SOMxVQTAM**	17,70%
14	CI Benchmark - Naive MLP (Crone)	17,84%
–	**RSOM**	17,94%
15	de Vos	18,24%
16	Yan	18,58%
17	CI Benchmark - Naive SVR (Crone, Pietsch)	18,60%
18	C49	18,72%
19	Perfilieva, Novak, Pavliska, Dvorak, Stepnicka	18,81%
20	Kurogi, Koyama, Tanaka, Sanuki	19,00%
21	Stat. Contender - Beadle	19,14%
22	Stat. Contender - Lewicke	19,17%
23	Sorjamaa, Lendasse	19,60%
24	Isa	20,00%
25	C28	20,54%
26	Duclos-Gosselin	20,85%
–	**SOMxRSOM**	21,64%
27	Stat. Benchmark - Naive	22,69%
28	Papadaki, Amaxopolous	22,70%
29	Stat. Benchmark - Hazarika	23,72%
30	C17	24,09%
31	Stat. Contender - Njimi, Melard	24,90%
32	Pucheta, Patino, Kuchen	25,13%
33	Corzo, Hong	27,53%

3.1 Example of learning on one of the samples

For example, take one of the 111 samples and see the results of the identification done by all four types of the described neural networks. The original sample is shown in Figure 3, the test 18 points of the

sample are separated from the learning sample with a vertical dashed line.

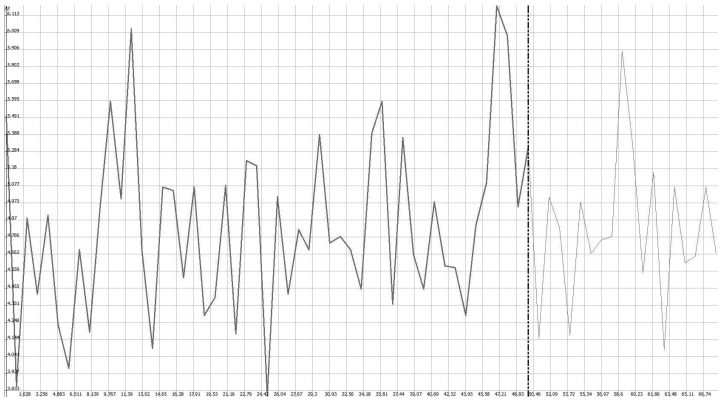

Figure 3. The original sample divided into learning and test samples with a vertical dashed line

In Figure 4 the results of testing VQTAM network on the last 18 points of the sample are overlain on the original sample. The SMAPE of 7.54% has been observed.

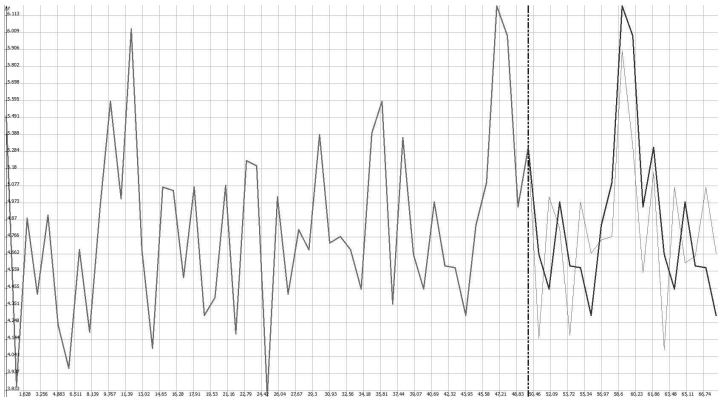

Figure 4. Results of testing VQTAM network, SMAPE 7.54%

In Figure 5 the results of testing RSOM network are overlain on

the original sample. The SMAPE of 7.79% has been observed.

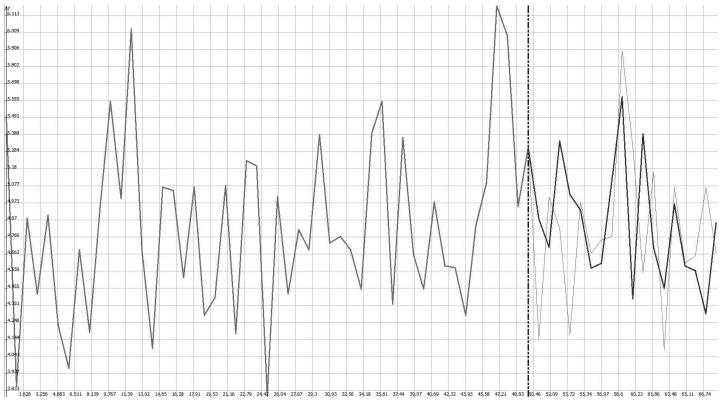

Figure 5. Results of testing RSOM network, SMAPE 7.79%

In Figure 6 the results of testing SOMxVQTAM modular network are presented. The SMAPE of 6.12% has been reached.

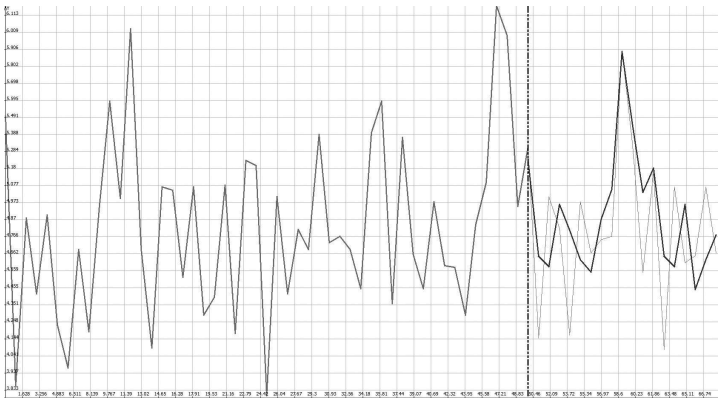

Figure 6. Results of testing SOMxVQTAM modular network, SMAPE 6.12%

In Figure 7 the results of testing SOMxRSOM modular network are overlain on the original sample. The SMAPE of 9.20% has been observed.

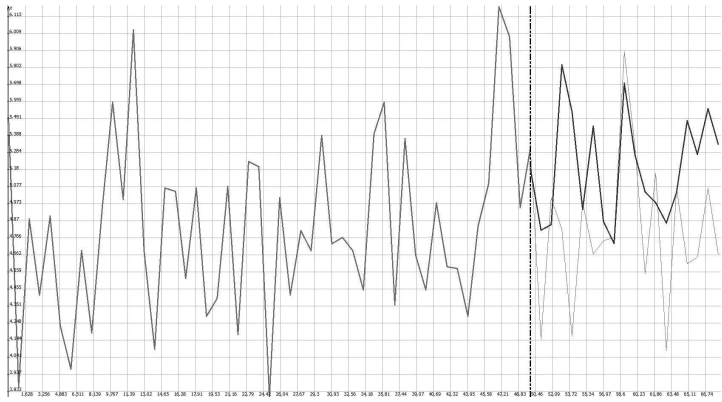

Figure 7. Results of testing SOMxRSOM modular network, SMAPE 9.20%

This and other tests that were concluded lead to a suggestion that modular modification gives a significant increase if accuracy in case of VQTAM modules, but modular networks are more sensitive to changes of learning parameters. The higher SMAPE in case of the SOMxRSOM modular network compared to RSOM network can be explaned by the fact that RSOM itself contains local models and during the learning process of the SOMxRSOM network those local models are treated as weight vectors for the whole SOMxRSOM network, but the local models are constructed for different parts of the input data and the local model corresponding to one of the neurons of one of the RSOMs are most likely to be constructed for the different part of the input data compared to the local model contained in another RSOM for the neuron on the same position.

4 Conclusion

In this article several SOM-based neural networks, that can be successfully used for dynamic object identification, have been described. Also, this study proves the necessity of investigating the possibilities of modular neural networks of higher complexity for identification of

dynamic objects and study of the ability of such networks to identify patterns in time series. Also it is necessary to develop an algorithm to automatically select learning parameters of presented in this paper neural networks in order to reduce prediction error.

References

[1] S. Haykin. *Neural Networks – A Comprehensive Foundation, 2nd Edition.* Prentice Hall International, Inc. (1998).

[2] N. Efremova, N. Asakura, T. Inui. *Natural object recognition with the view-invariant neural network.* The Proceedings of the 5th International Conference of Cognitive Science (CoSci), (2012).

[3] A. Trofimov, I. Povidalo, S. Chernetsov. *Usage of the self-learning neural networks for the blood glucose level of patients with diabetes mellitus type 1 identification.* Science and education. vol. 5 (2010), (In Russian). http://technomag.edu.ru/doc/142908.html

[4] T. Koskela. *Neural network methods in analyzing and modelling time varying processes.* Espoo. (2003), pp. 1–72.

[5] Luis Gustavo M. Souza, Guilherme A. Barreto. *Multiple Local ARX Modeling for System Identification Using the Self-Organizing Map.* Proceedings, European Symposium on Artificial Neural Networks – Computational Intelligence and Machine Learning. Bruges (Belgium). (2010).

[6] M. Varsta, J. Heikkonen. *A recurrent Self-Organizing Map for temporal sequence processing.* Springer. (1997), pp. 421–426.

[7] A. Lotfi, J. Garibaldi. *In Applications and Science in Soft Computing, Advances in Soft Computing Series.* Springer. (2003), pp. 3–8.

[8] K. Tokunaga, T. Furukawa. *SOM of SOMs.* Neural Networks. vol. 22 (2009), pp. 463–478.

[9] K. Tokunaga, T. Furukawa. *Modular network SOM.* Neural Networks. vol. 22 (2009), pp. 82–90.

[10] N. K. Logothetis, J. Pauls, T. Poggiot. *Shape representation in the inferior temporal cortex of monkeys.* Current Biology. vol. 5 (1995), pp. 552–563.

[11] T. Vetter, A. Hurlbert, T. Poggio. *View-based Models of 3D Object Recognition: Invariance to Imaging Transformations.* Cerebral Cortex. vol. 3 (1995), pp. 261–269.

[12] K. Aihara, G. Matsumoto, M. Ichikawa. *An alternating periodic-chaotic sequence observed in neural oscillators.* Phys. Lett. A. vol. 111(5) (1985), pp. 251–255.

[13] K. Aihara, T. Takabe, M. Toyoda. *Chaotic neural networks.* Phys. Lett. A. vol. 144(6/7) (1990), pp. 333–340.

[14] *Artificial Neural Network & Computational Intelligence Forecasting Competition.* (2007). http://www.neural-forecasting-competition.com/NN3/results.htm

Aleksey Averkin, Veaceslav Albu,
Sergey Ulyanov, Ilya Povidalo

Received December 2, 2013

Aleksey Averkin
Institution of Russian Academy of Sciences Dorodnicyn Computing Centre of RAS
Vavilov st. 40, 119333 Moscow, Russia
E–mail: averkin2003@inbox.ru

Veaceslav Albu
Institute of Mathematics and Computer Science
Academiei 5, Kishinev, MD 2028 Moldova
E–mail: vaalbu@googlemail.com

Sergey Ulyanov
International University of Nature, Society and Man "Dubna"
Universitetskaya st. 19, 141980 Dubna, Moscow region, Russia
E–mail: ulyanovsv@mail.ru

Ilya Povidalo
International University of Nature, Society and Man "Dubna"
Universitetskaya st. 19, 141980 Dubna, Moscow region, Russia
E–mail: ipovidalo@gmail.com